La duración del examen será de 2:00 h.
El problema 1 vale un 40% de la nota final. Los problemas 2 y 3 un 30% cada uno.

Problema 1: (Nota: los apartados 1.1 y 1.2 son independientes)

1.1 Queremos hallar el polinomio de Hermite h(t) que interpola a la función $f(t) = \sin(t)$ en los puntos $t_0 = 0$ y $t_1 = \pi$.

a) Rellenar la siguiente tabla con los datos necesarios para la interpolación:

t_k	0	π
$f(t_k)$		
$f'(t_k)$		

¿De qué orden será el polinomio resultante? Justificar

b) Construir el correspondiente triángulo de diferencias divididas (generalizadas).

c) Dar la expresión del polinomio de h(t) a partir de las diferencias divididas.

d) Hallar una cota del error de interpolación $e(t) = |f(t) - h(t)|$ en el intervalo $[0, \pi]$.

1.2 Queremos hallar el polinomio de segundo grado, $p(t) = a + bt + ct^2$ que verifique la condición p'(-1) = 0 y que mejor ajuste (en el sentido de mínimos cuadrados) los datos de la siguiente tabla:

t_k	-1	0	1
y_k	1	1	-1

a) Dar la expresión más general de un polinomio de grado 2 que verifique p'(-1)=0.

b) Plantear (sin resolver) el sistema sobredeterminado al que llegamos para resolver el problema de ajuste.

Problema 2: Sea la ecuación

$$e^{-x} - \frac{1}{3} = 0$$

Se pide:

- Comprobar que tiene una raíz s en el intervalo $[a,b]=[1,2]$.
- Se quiere calcular un valor aproximado de la raíz s utilizando el método de Newton, a partir de un punto inicial x_0. Calcular un $x_0 \in [a,b]$ para el cual el método es convergente. Demostrarlo.
- A partir del valor x_0 obtenido, calcular la primera iteración del método de Newton x_1.
- A partir del valor x_0 obtenido, calcular el número de iteraciones necesarias para obtener una aproximación de la solución s con un error menor que la precisión de la máquina (coma flotante en doble precisión).

Problema 3:

La solución exacta del sistema $\mathrm{Ax}=\mathrm{b}$ dado por:

$$\begin{pmatrix} 7 & 10 \\ 5 & 7 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.7 \end{pmatrix}$$

es $x_1 = 0$ y $x_2 = 0.1$

Se pide:

a) Calcular la factorización LU de la matriz A, y a partir de ella calcular A^{-1}.

b) Se define el condicionamiento de una matriz A como $c(A) = \|A\|\|A^{-1}\|$.
Estimar el condicionamiento para la matriz A del sistema anterior.

c) Resolver el sistema perturbado:

$$\begin{pmatrix} 7 & 10 \\ 5 & 7 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1.01 \\ 0.69 \end{pmatrix}$$

y calcular la perturbación relativa de su solución. Justificar dicho resultado a partir del condicionamiento de la matriz A del sistema.

Problema 1:

1.1 a) Rellenar la siguiente tabla con los datos necesarios para la interpolación:

t_k	0	π
$f(t_k)=\sin(t_k)$	0	0
$f'(t_k)=\cos(t_k)$	1	-1

4 condiciones -> 4 coeficientes -> polinomio grado 3

b) Construir tabla de diferencias divididas (generalizadas) y dar la expresión de h(t) a partir de las diferencias divididas

Se trata de plantear la tabla de diferencias divididas repitiendo aquellos puntos donde nos dan valor de la función y derivada. En el caso de Hermite nos dan función y derivada en cada punto, así que repetiremos cada punto (t=0 y t=pi) dos veces. En los casos en los que la definición ordinaria de diferencia dividida daría 0/0 usamos la generalización de las diferencias divididas: $f[t_0,t_0]=f'(t_0)$

t_k	$f[t_0]$	$f[t_0,t_0]$	$f[t_0,t_0,t_1]$	$f[t_0,t_0,t_1,t_1]$
0	0	$f'(0)=\cos(0)=1$	$\frac{0-1}{\pi}=-\frac{1}{\pi}$	$\frac{-1/\pi-(-1/\pi)}{\pi}=0$
0	0	$\frac{0-0}{\pi-0}=0$	$\frac{-1-0}{\pi}=-\frac{1}{\pi}$	
π	0	$f'(\pi)=\cos(\pi)=-1$		
π	0			

Las diferencias divididas resultan ser (primera fila) 0 , 1 , $-\frac{1}{\pi}$ y 0.

Por lo tanto el polinomio de interpolación de Hermite es:

$$h(t)=0+1(t-0)-\frac{1}{\pi}(t-0)^2+0(t-0)^2(t-1)=t-\frac{t^2}{\pi}$$

d) Hallar una cota del error de interpolación $e(t)=|f(t)-h(t)|$ en el intervalo $[0,\pi]$.

Cota del error: $e(t)=\frac{f^{IV}(\zeta)}{4!}(t-\pi)^2t^2$

Las derivadas de f=sin(t) serán o sen o cos, todas ellas acotadas por 1:

$$e(t)\le\frac{1}{4!}(t-\pi)^2t^2$$

El segundo término tendrá su máximo (dentro del intervalo) en el punto medio $t=\pi/2$ (si no se ve claro se comprueba derivando) y el mayor valor posible de e(t) es:

$$e(t)\le\frac{1}{4!}\left(\frac{\pi}{2}\right)^2\left(\frac{\pi}{2}\right)^2\approx 0.25$$

1.2 Queremos hallar el polinomio de segundo grado, $p(t) = a + bt + ct^2$ que verifique la condición p'(-1) = 0 y que mejor ajuste (en el sentido de mínimos cuadrados) los datos de la siguiente tabla:

t_k	-1	0	1
y_k	1	1	-1

- Dar la expresión más general de un polinomio de grado 2 que verifique p'(-1)=0.

$$p'(t) = b + 2ct \quad \Rightarrow \quad p'(-1) = b - 2c = 0 \quad \Rightarrow \quad b = 2c$$

luego la expresión de p(t) queda como $p(t) = a + 2ct + ct^2 = a + c(t^2 + 2t)$

- Plantear (sin resolver) el sistema sobredeterminado al que llegamos para resolver el problema de ajuste.

El problema queda como uno de ajustar los datos (y_k) en un espacio de 2 dimensiones, usando la base $\{1, t^2 + 2t\}$. El sistema sobredeterminado a resolver es pues:

$$\begin{pmatrix} 1 & t_0^2 + 2t_0 \\ 1 & t_1^2 + 2t_1 \\ 1 & t_2^2 + 2t_2 \end{pmatrix} \begin{pmatrix} a \\ c \end{pmatrix} \approx \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} a \\ c \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$$

Problema 2: Sea la función $f(x) = e^{-x} - \frac{1}{3}$ continua.

a) $f(1)f(2) = 0.03.(-0.19) < 0$ la función cambia de signo en el intervalo $[a,b] = [1,2]$ y por tanto f(x) tiene una raíz $s \in [1,2]$.

b) $f'(x) = -e^{-x}, \quad f''(x) = e^{-x}$

$$\max_{x\in[1,2]} |f''(x)| = \max_{x\in[1,2]} e^{-x} = e^{-1}, \quad \min_{x\in[1,2]} |f'(x)| = \min_{x\in[1,2]} e^{-x} = e^{-2} \quad \Rightarrow \quad M = \frac{\max_{x\in[1,2]} |f''(x)|}{2 \min_{x\in[1,2]} |f'(x)|} = \frac{e}{2} \cong 1.36 .$$

Si tomamos el punto $x_0 = 1.5$ tenemos que $e_0 = |s - x_0| \le 0.5$. Para ese valor de x_0 se verifica $e_0 \le \frac{1}{M} = \frac{2}{e} \cong 0.73$. Podemos concluir que el método de Newton es convergente para $x_0 = 1.5$, ya que $e_0 \le \frac{1}{M}$.

Las iteraciones del método de Newton tienen la forma

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{\exp(x_n) - 1/3}{-\exp(x_n)} = x_n + 1 - \frac{\exp(x_n)}{3}$$

$x\in[1,2]$

Si arrancamos en $x_0 = 1.5$ la primera iteración es

$$x_1 = 1.5 + 1 - \frac{\exp(1.5)}{3} \cong 1.0061$$

$x \in [1,2]$

La cota del error del método de Newton viene dada por la expresión

$$e_n \le \frac{1}{M}(Me_0)^{2^n} = \frac{2}{e}\left(\frac{e}{2}\frac{1}{2}\right)^{2^n} = \frac{2}{e}\left(\frac{e}{4}\right)^{2^n} \le 10^{-15}$$

$x \in [1,2]$

Donde se ha sustituido el valor de $M = \frac{e}{2}$ y la cota $e_0 \le 0.5$. Operando y tomando logaritmos en dos ocasiones obtenemos

$$n \ge \log\left[\log\left(\frac{e}{2}10^{-15}\right) / \log\left(\frac{e}{4}\right)\right] / \log(2) \cong 6.43$$

$x \in [1,2]$

Luego, arrancando en $x_0 = 1.5$, con n=7 iteraciones obtenemos la solución con la precisión de la máquina (10^{-15}).

Problema 3:

a) Hacemos la descomposición o factorización $A = LU$

$$A = \begin{pmatrix} 7 & 10 \\ 5 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix}\begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} = LU$$

$a_{11} = 7 = u_{11}$

$a_{12} = 10 = u_{12}$

$a_{21} = 5 = l_{21}u_{11} \rightarrow l_{21} = \frac{5}{7}$

$a_{22} = 7 = l_{21}u_{12} + u_{22} \rightarrow u_{22} = -\frac{1}{7}$

Por tanto: $L = \begin{pmatrix} 1 & 0 \\ \frac{5}{7} & 1 \end{pmatrix}$ y $U = \begin{pmatrix} 7 & 10 \\ 0 & -\frac{1}{7} \end{pmatrix}$

Para calcular la 1ª columna de A^{-1}, resolvemos el sistema $Ax = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ y para calcular los elementos de la 2ª columna, resolvemos $Ax = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ con $A = LU$

- $L(Ux) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, haciendo $Ux = y$, tenemos $Ly = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$Ly = \begin{pmatrix} 1 & 0 \\ \frac{5}{7} & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, sistema triangular inferior, cuya solución es:

$y_1 = 1\,;\ y_2 = -\frac{5}{7}$

Resolviendo ahora $Ux = y$, es decir, en nuestro caso $\begin{pmatrix} 7 & 10 \\ 0 & -\frac{1}{7} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -\frac{5}{7} \end{pmatrix}$, sistema triangular superior, cuya solución es:

$x_1 = -7\,;\ x_2 = 5$

- $L(Ux) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, operando de la misma forma tiene como solución:

$x_1 = 10\,;\ x_2 = -5$

Con lo que:

$$A^{-1} = \begin{pmatrix} -7 & 10 \\ 5 & -7 \end{pmatrix}$$

b) Trabajando, por ejemplo, con la norma 1 , el condicionamiento es : $c(A) = \|A\|_1 \|A^{-1}\|_1 = 17 \cdot 17 = 289$
que indica que la matriz está mal condicionada (el condicionamiento óptimo es 1)

c) Resolvemos el sistema aprovechando la descomposición $A = LU$ puesto que la matriz de ambos sistemas es la misma variando ligeramente sus términos independientes.

$L(Ux^*) = \begin{pmatrix} 1{,}01 \\ 0{,}69 \end{pmatrix}$

$Ly = \begin{pmatrix} 1 & 0 \\ \frac{5}{7} & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1{,}01 \\ 0{,}69 \end{pmatrix} \rightarrow y_1 = 1{,}01;\ y_2 = -0{,}03$

$Ux = \begin{pmatrix} 7 & 10 \\ 0 & -\frac{1}{7} \end{pmatrix}\begin{pmatrix} x_1^* \\ x_2^* \end{pmatrix} = \begin{pmatrix} 1{,}01 \\ -0{,}03 \end{pmatrix} \rightarrow x_1^* = -0{,}16;\ x_2^* = 0{,}21$

La perturbación relativa es:

$\frac{\|x - x^*\|_1}{\|x\|_1} = 2{,}7$ mayor del 100% lo que se justifica por el mal condicionamiento de la matriz A.

Ejercicio 1: Se desea **interpolar** la siguiente tabla de datos:

t_k	0.5	1.5	2.5	3.5	4.5
y_k	0.35	0.48	0.85	0.96	0.56

con un polinomio p(t) del grado adecuado. Generar la matriz H1 del correspondiente sistema lineal y resolver:

1.1 Adjuntar código usado. Listar coeficientes del polinomio interpolador.

Repetimos la interpolación usando funciones u(t) del tipo:

$$u(t) = A\cos(t) + B\sin(t) + C + Dte^{-t} + \frac{E}{1+t^2}.$$

Generar la nueva matriz H2 y resolver.

1.2 Código para generar la matriz H2. Resolver y listar los coeficientes de u(t)

Evaluar ambas funciones interpoladoras en el intervalo [0,5] y hacer una gráfica superponiendolas sobre los puntos de la tabla (dibujados como puntos aislados de color rojo). Usar línea azul para p(t) y verde para u(t).

1.3 Adjuntar código y gráfica obtenida.

Calcular el condicionamiento de las matrices H1 y H2 (usar la función cond de MATLAB). ¿Qué conjunto de coeficientes (los de p(t) o los de u(t)) podemos esperar que hayan sido calculados con un menor error? Justificar.

1.4 Resultados. Responder a la pregunta y justificar respuesta

Verificarlo calculando la discrepancia entre los datos de la tabla (y_k) y los correspondientes valores de $p(t_k)$ y $u(t_k)$. ¿Se confirman las expectativas?

1.5 Código, Listado de discrepancias. Responder a la pregunta

Finalmente queremos **ajustar los datos** de la tabla anterior con una función de la forma $f(t) = \dfrac{1}{A + B\cos(t)}$.

Plantear el sistema lineal a resolver para hallar los coeficientes A y B.

1.6 Código para resolver el problema de ajuste. Coeficientes A y B encontrados

Superponer sobre la gráfica anterior la nueva curva de ajuste (intervalo [0,5]).

1.7 Adjuntar código y gráfica obtenida.

Calcular los residuos de ajuste, esto es, la diferencia entre los datos (y_k) y los resultados del ajuste en los puntos dados ($f(t_k)$)

1.8 Código y residuos obtenidos.

Ejercicio 2: Sea la función

$$R(x) = \frac{r}{1 + xK}$$

Aplicamos la iteración de punto fijo $x_{k+1} = R(x_k) \quad k = 0, 1, 2, ...$ y obtenemos la sucesión

$$x_{k+1} = \frac{r}{1 + x_k K}$$

a) Escribir un script para calcular las 10 primeras iteraciones de la fórmula anterior, para los valores de $r = 3, K = 1$ y arrancando a partir de $x_0 = 1$.

```
2.a % Escribir aquí el código.
    % Volcar los resultados.
    % ¿Converge la sucesión?. En su caso, ¿a que valor?.
```

b) Dibujar una gráfica de la función $R(x)$ en el intervalo [0,3] (línea azul 'b'), en cada iteración dibujar el punto $(x_k, R(x_k))$ (cuadrados rojos 'rs'). En la última iteración dibujar el punto de color verde ('gs').

```
2.b  Escribir aquí el código.
     Insertar la gráfica.
```

c) Repetir el apartado a) iterando 70 veces. Consideramos como valor exacto de la raíz el valor de la última iteración, esto es, $s = x_{70}$. Calcular el número de cifras significativas de precisión obtenidas en cada iteración. Dibujar en una gráfica el número de iteración y el número de cifras significativas de precisión obtenidas en esa iteración. ¿Cuál es la velocidad de convergencia (lineal, cuadrática, ...)?

```
2.c % Insertar el código y volcar los resultados.
    % Insertar la gráfica.
    % ¿Cuántas iteraciones son necesarias para alcanzar la precisión de la
      máquina en doble precisión?.
    % Velocidad de convergencia (lineal, cuadrática,…).
```

d) Repetir el apartado anterior para la función

$$S(x) = \frac{r}{1 + (x/K)^2}$$

para los valores de $r = 3, K = 1$ y arrancando a partir de $x_0 = 1$.

```
2.d % Insertar el código, los resultados y la gráfica.
```

Ejercicio 3: Escribir una rutina que reciba una matriz cuadrada A de tamaño N x N, y devuelva dos matrices L (triangular inferior) y U (triangular superior), de forma que A=LU. La función **debe seguir el siguiente algoritmo**:

1) Inicializar la diagonal de U con unos.
2) Para k=1, ..., N, hacer

$$\begin{cases} \text{Calcular } L_{kk} = A_{kk} - \sum_{m=1}^{k-1} L_{km} U_{mk} \\ \text{Para r desde } k+1 \text{ hasta N calcular: } \begin{cases} U_{kr} = \dfrac{\left(A_{kr} - \sum_{m=1}^{k-1} L_{km} U_{mr}\right)}{L_{kk}} \\ L_{rk} = A_{rk} - \sum_{m=1}^{k-1} L_{rm} U_{mk} \end{cases} \end{cases}$$

Verificar que vuestra rutina funciona aplicándola a la matriz:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 6 \\ 3 & 6 & 12 \end{pmatrix} \text{ que debe dar como factorización } L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & 3 & 3 \end{pmatrix} \text{ y } U = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

3.1 Adjuntar código de la función resultante.

Comprobar el error cometido en vuestra rutina para matrices grandes. Para ello, crear una matriz de tamaño 200 con el siguiente comando:

```
>> N=200; A = rand(N)+5*N*eye(N);
```

y verificar la diferencia entre A y el producto LU. Usar alguna norma matricial (función norm de MATLAB) para estimar el orden de las discrepancias.

3.2 Adjuntar código y volcado de resultados.

Finalmente, comparar los resultados de vuestra rutina con la rutina lu de Matlab. Crear una matriz de tamaño N=5 con el comando anterior y comparar vuestra factorización con la obtenida con MATLAB:

```
[L_m U_m] = lu(A)
```

3.3 ¿Son iguales los resultados? ¿Qué relación puede haber entre ambos resultados? Justificar y comentar.

SOLUCION PROBLEMA 1

```
tk=[0.5:1:4.5]';
yk =[ 0.35 0.48 0.85 0.96 0.56]';

H1 = [tk.^4 tk.^3 tk.^2 tk tk.^0]; c1=H1\yk   %Polinomial H1,c1
H2=[cos(tk) sin(tk) tk.^0 tk.*exp(-tk) 1./(1+tk.^2)];  c2=H2\yk

% Evaluación y gráficas
tt=[0.5:0.001:5];
pp=polyval(c1,tt);
uu=c2(1)*cos(tt)+c2(2)*sin(tt)+c2(3)+c2(4)*tt.*exp(-tt)+c2(5)./(1+tt.^2);
plot(tk,yk,'ro',tt,pp,'b',tt,uu,'g');

% Cond y discrepancias
fprintf('Cond de H1 = %.0f\nCond de H2 %.0f\n',cond(H1),cond(H2));
disc_1=yk-H1*c1,   disc_2=yk-H2*c2,

% Ajuste
fk=1./yk;
H=[tk.^0 cos(tk)];
c = H\fk; A=c(1), B=c(2),

ff=1./(A+B*cos(tt));
hold on; plot(tk,yk,'ro',tt,ff,'r:'); hold off

r = yk - 1./(A+B*cos(tk))  % Residuos
```

VOLCADO

```
c1 =    0.0104   -0.1667    0.7190   -0.8183    0.5996
c2 =   -0.6005   -0.1644    0.1993    0.8999    0.6045
```

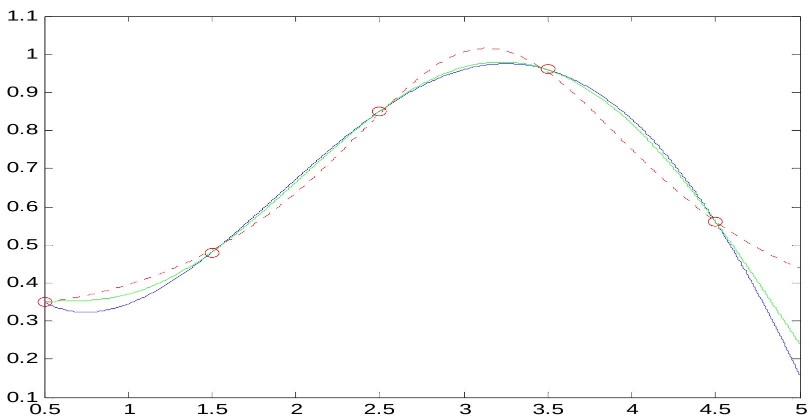
Cond de H1 = 8893   Cond de H2 = 104   El condicionamiento mayor de H1 indica mayores errores en los coeficientes del polinomio, lo que se refleja en las siguientes discrepancias:

```
disc_1 =  1.0e-015 * [ -0.0555 -0.2220  -0.1110 -0.8882  -0.4441]
disc_2 =  1.0e-016 * [  0.5551       0        0       0        0]

A =   1.9899  B =   1.0051

res =   0.0018   -0.0052    0.0059    0.0064   -0.0024
```

GRAFICA



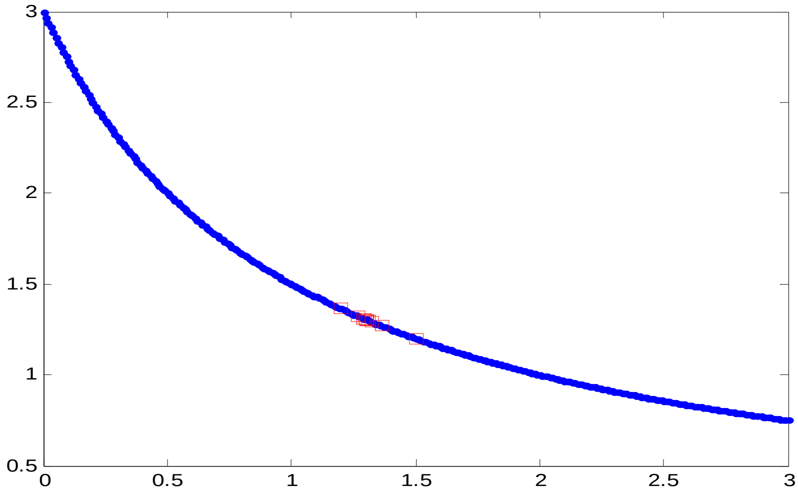
Solución PROBLEMA 2

Apdos. a) y b)

```
r=3,K=1,x=1,N=10
y=0:0.01:3;Ry=r./(1+y*K);plot(y,Ry,'b.'); hold on;
for k=1:N
    x=r/(1+x*K);Rx=r/(1+x*K);
    fprintf('Iteración: %d valor x_k: %.16f \n',k,x);
    plot(x,Rx,'sr');
end
```

```
r = 3
K =1
x =1
N =10

Iteración: 1 valor x_k: 1.5000000000000000
Iteración: 2 valor x_k: 1.2000000000000000
Iteración: 3 valor x_k: 1.3636363636363635
Iteración: 4 valor x_k: 1.2692307692307694
Iteración: 5 valor x_k: 1.3220338983050848
Iteración: 6 valor x_k: 1.2919708029197081
Iteración: 7 valor x_k: 1.3089171974522293
Iteración: 8 valor x_k: 1.2993103448275862
Iteración: 9 valor x_k: 1.3047390521895621
Iteración: 10 valor x_k: 1.3016657990629881
```

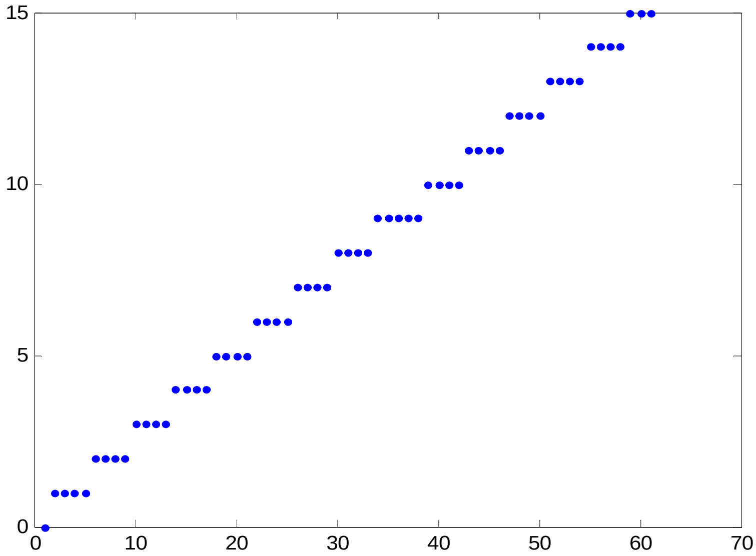


El método converge al valor s=1.30

Apdo c)

```
r=3,K=1,x=1,N=70
y=0:0.01:3;Ry=r./(1+y*K);
for k=1:N
    x=r/(1+x*K);Rx=r/(1+x*K);
    fprintf('Iteración: %d valor x_k: %.16f \n',k,x);
end
s=x,r=3,K=1;x=1,
for k=1:N
    x=r/(1+x*K);fx=0;Erel=abs(x-s)/abs(s);N_cifras(k)=floor(-
log10(Erel));Rx=r/(1+x*K);
    fprintf('Iteración: %d valor x_k: %.16f N. cifras %3d E_rel
%.3e\n',k,x,N_cifras(k),Erel);
end

plot([1:N],N_cifras,'.')
```



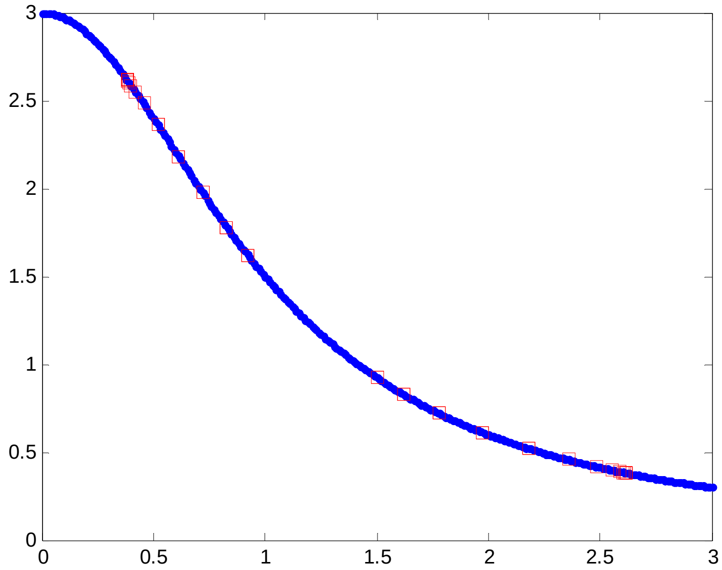
La precisión de la máquina se alcanza en la iteración 59.
La velocidad de convergencia es lineal: necesita 4 iteraciones para aumentar una unidad las cifras significativas de precisión.

Apdo d)

```
clear all
r=3,K=1,s=1;x=1,N=70
y=0:0.01:3;Sy=r./(1+(y/K).^2);plot(y,Sy,'b.'); hold on;
for k=1:N
     x=r/(1+(x/K)^2);fx=0;Erel=abs(x-s)/abs(s);Sx=r/(1+(x/K)^2);
     fprintf('Iteración: %d valor x_k: %.16f Erel %.3e\n',k,x,Erel);
     plot(x,Sx,'sr');
end
```

```
r =     3
K =    1
x =    1
N =    70
Iteración: 1 valor x_k: 1.5000000000000000 Erel 5.000e-001
Iteración: 2 valor x_k: 0.9230769230769231 Erel 7.692e-002
Iteración: 3 valor x_k: 1.6198083067092650 Erel 6.198e-001
Iteración: 4 valor x_k: 0.8278650659966538 Erel 1.721e-001
Iteración: 5 valor x_k: 1.7800345266498552 Erel 7.800e-001
Iteración: 6 valor x_k: 0.7196793829387234 Erel 2.803e-001
Iteración: 7 valor x_k: 1.9763647667666466 Erel 9.764e-001
Iteración: 8 valor x_k: 0.6114939221091391 Erel 3.885e-001
Iteración: 9 valor x_k: 2.1835255927895489 Erel 1.184e+000
Iteración: 10 valor x_k: 0.5201304335473228 Erel 4.799e-001
Iteración: 11 valor x_k: 2.3612088001854512 Erel 1.361e+000
Iteración: 12 valor x_k: 0.4562524610454081 Erel 5.437e-001
Iteración: 13 valor x_k: 2.4831018541187206 Erel 1.483e+000
Iteración: 14 valor x_k: 0.4186555820010942 Erel 5.813e-001
Iteración: 15 valor x_k: 2.5525995114671631 Erel 1.553e+000
Iteración: 16 valor x_k: 0.3991610026400729 Erel 6.008e-001
Iteración: 17 valor x_k: 2.5877026198330175 Erel 1.588e+000
Iteración: 18 valor x_k: 0.3898025142236674 Erel 6.102e-001
Iteración: 19 valor x_k: 2.6042887424864980 Erel 1.604e+000
Iteración: 20 valor x_k: 0.3854891672647959 Erel 6.145e-001
Iteración: 21 valor x_k: 2.6118710103289007 Erel 1.612e+000
Iteración: 22 valor x_k: 0.3835399889074168 Erel 6.165e-001
Iteración: 23 valor x_k: 2.6152840687702996 Erel 1.615e+000
Iteración: 24 valor x_k: 0.3826671788923003 Erel 6.173e-001
Iteración: 25 valor x_k: 2.6168096511984809 Erel 1.617e+000
Iteración: 26 valor x_k: 0.3822779631336261 Erel 6.177e-001
Iteración: 27 valor x_k: 2.6174894126279233 Erel 1.617e+000
```

```
Iteración: 28 valor x_k: 0.3821047202266927 Erel 6.179e-001
Iteración: 29 valor x_k: 2.6177918706417103 Erel 1.618e+000
Iteración: 30 valor x_k: 0.3820276722769306 Erel 6.180e-001
Iteración: 31 valor x_k: 2.6179263640451249 Erel 1.618e+000
Iteración: 32 valor x_k: 0.3819934186393521 Erel 6.180e-001
Iteración: 33 valor x_k: 2.6179861522567274 Erel 1.618e+000
Iteración: 34 valor x_k: 0.3819781928036208 Erel 6.180e-001
Iteración: 35 valor x_k: 2.6180127274361076 Erel 1.618e+000
Iteración: 36 valor x_k: 0.3819714253710091 Erel 6.180e-001
Iteración: 37 valor x_k: 2.6180245391486681 Erel 1.618e+000
Iteración: 38 valor x_k: 0.3819684175452471 Erel 6.180e-001
Iteración: 39 valor x_k: 2.6180297889017572 Erel 1.618e+000
Iteración: 40 valor x_k: 0.3819670807184024 Erel 6.180e-001
Iteración: 41 valor x_k: 2.6180321221457112 Erel 1.618e+000
Iteración: 42 valor x_k: 0.3819664865700969 Erel 6.180e-001
Iteración: 43 valor x_k: 2.6180331591470454 Erel 1.618e+000
Iteración: 44 valor x_k: 0.3819662225035828 Erel 6.180e-001
Iteración: 45 valor x_k: 2.6180336200373220 Erel 1.618e+000
Iteración: 46 valor x_k: 0.3819661051405688 Erel 6.180e-001
Iteración: 47 valor x_k: 2.6180338248776018 Erel 1.618e+000
Iteración: 48 valor x_k: 0.3819660529792058 Erel 6.180e-001
Iteración: 49 valor x_k: 2.6180339159177568 Erel 1.618e+000
Iteración: 50 valor x_k: 0.3819660297963733 Erel 6.180e-001
Iteración: 51 valor x_k: 2.6180339563800539 Erel 1.618e+000
Iteración: 52 valor x_k: 0.3819660194928913 Erel 6.180e-001
Iteración: 53 valor x_k: 2.6180339743632985 Erel 1.618e+000
Iteración: 54 valor x_k: 0.3819660149135657 Erel 6.180e-001
Iteración: 55 valor x_k: 2.6180339823558518 Erel 1.618e+000
Iteración: 56 valor x_k: 0.3819660128783099 Erel 6.180e-001
Iteración: 57 valor x_k: 2.6180339859080979 Erel 1.618e+000
Iteración: 58 valor x_k: 0.3819660119737517 Erel 6.180e-001
Iteración: 59 valor x_k: 2.6180339874868741 Erel 1.618e+000
Iteración: 60 valor x_k: 0.3819660115717258 Erel 6.180e-001
Iteración: 61 valor x_k: 2.6180339881885519 Erel 1.618e+000
Iteración: 62 valor x_k: 0.3819660113930478 Erel 6.180e-001
Iteración: 63 valor x_k: 2.6180339885004091 Erel 1.618e+000
Iteración: 64 valor x_k: 0.3819660113136352 Erel 6.180e-001
Iteración: 65 valor x_k: 2.6180339886390125 Erel 1.618e+000
Iteración: 66 valor x_k: 0.3819660112783407 Erel 6.180e-001
Iteración: 67 valor x_k: 2.6180339887006134 Erel 1.618e+000
Iteración: 68 valor x_k: 0.3819660112626543 Erel 6.180e-001
Iteración: 69 valor x_k: 2.6180339887279920 Erel 1.618e+000
Iteración: 70 valor x_k: 0.3819660112556826 Erel 6.180e-001
```



Como se puede observar, en este caso el método iterativo no converge.

Solución PROBLEMA 3

Apdo 3.1

```
function [L,U]=factorLU(A)

L=zeros(size(A));U=eye(size(A,1));
n=size(A,1);

for k=1:n
  L(k,k)=A(k,k) - sum(L(k,1:k-1).*U(1:k-1,k)');
  for l=k+1:n
    U(k,l)=(A(k,l) - sum(L(k,1:k-1).*U(1:k-1,l)'))/L(k,k);
    L(l,k)=(A(l,k) - sum(L(l,1:k-1).*U(1:k-1,k)'));
  end
end
return
```

Apdo 3.2

```
N=200; A = rand(N)+5*N*eye(N);
[L U]=factorLU(A);
error200=norm(A-L*U,1)  % del orden de 1e-13
```

Apdo 3.3

```
N=5; A = rand(N)+5*N*eye(N)
[L U]=factorLU(A),
[L_m U_m]=lu(A),
```

```
L =   25.2924         0         0         0         0
       0.0919   25.2483         0         0         0
       0.2771    0.4237   25.1737         0         0
       0.0473    0.3953    0.1692   25.4010         0
       0.3184    0.0508    0.3589    0.9095   25.7193

U =    1.0000    0.0376    0.0195    0.0375    0.0034
            0    1.0000    0.0029    0.0034    0.0128
            0         0    1.0000    0.0323    0.0065
            0         0         0    1.0000    0.0064
            0         0         0         0    1.0000

L_m =  1.0000         0         0         0         0
       0.0036    1.0000         0         0         0
       0.0110    0.0168    1.0000         0         0
       0.0019    0.0157    0.0067    1.0000         0
       0.0126    0.0020    0.0143    0.0358    1.0000


U_m = 25.2924    0.9511    0.4926    0.9473    0.0855
            0   25.2483    0.0729    0.0850    0.3242
            0         0   25.1737    0.8120    0.1631
            0         0         0   25.4010    0.1632
            0         0         0         0   25.7193
```

Puesto que son dos factorizaciones triangular inferior por superior, la diferencia está en las diagonales. La diagonal de L por este algoritmo es la diagonal de U de MatLab.

Si llamamos D a la matriz diagonal que tiene como diagonal cualquiera de esas dos entonces L*inv(D) es la L que da Matlab y D*U es la U que da Matlab (puesto que quedan unos en la diagonal de L*inv(D) y la unicidad de la factorización).